Xiaoyun Zhi

In this coursework, a subset of CIFAR-10 dataset is used, namely 5,000 images for training and 1,000 images for testing. Data are normalized to the range of $[-1, 1]$ by using `transforms` in `torchvision`. The subset training data is almost equally distributed for 10 classes as shown in Figure 1. As there are minor variations between each class, evaluations including accuracy, f1, precision, recall, will be calculated by weighted average.

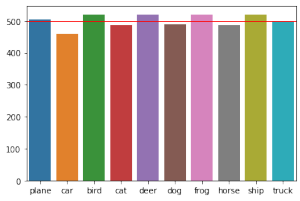



Figure 1: Data distribution over 10 classes

# 1 Task 1

In task 1, `PCA` method from `sklearn.decomposition` is used since it can receive a percentage of information to retain and return the corresponding reduced feature vectors. PCAs that can explain 10%, 30%, 50%, 70%, 100% amount of variance are fitted using training data, and subsequently they are used to transform testing data into difference reduced feature vectors.

Figure 2 shows the reconstructed image of a horse using `inverse_transform`, with difference explained variance at the top and its corresponding number of feature vectors at the bottom.

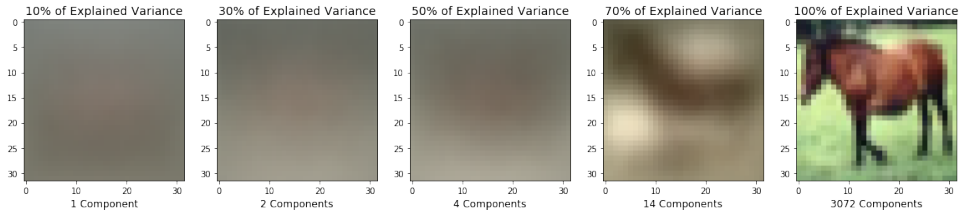



Figure 2: Explained variance vs. number of feature vectors (Image: horse)

# 2 Task 2

## 2.1 Task 2.1

In task 2.1, different input feature vectors from task 1 are evaluated on the basis of average validation accuracy using 10-fold cross validation of linear SVM, and `GridSearchCV` is used to search the best value for parameter `C`.

In Figure 3, it shows that parameter `C` with a value of 0.001 is best for this linear SVM in the range of $[10^{-4}, 1]$. In addition, the input vector with 3072 features (same as 100% information kept, original) has the best validation accuracy of 37.5% with C set to 0.001. This figure also suggests there might be better input features with dimension between 3072 (100%) and 14 (70%), which is possible to result in higher accuracy score and reduce computational time than 3072-dimentional input vectors.

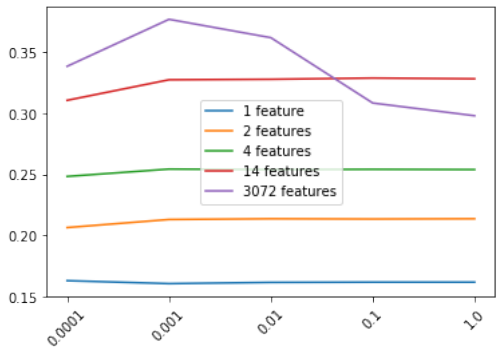
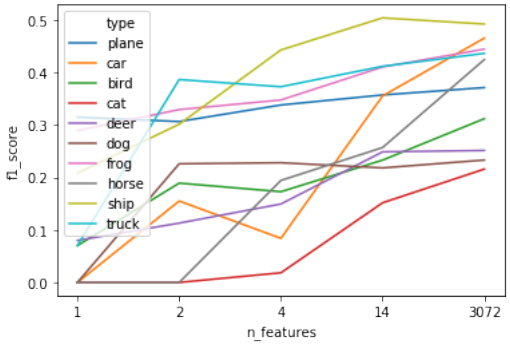



Figure 3: 10-fold cross validation accuracy with difference parameter C

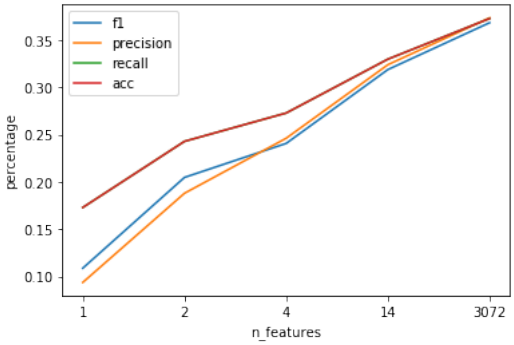
## 2.2 Task 2.2

From the cross validation results in task 2.1, the best parameter value C {`C=0.001`} is set for this subtask. Linear SVMs are refitted on the entire training set for different input feature vectors, and then tested on the 1,000 testing set, recording various testing results, such as F1, accuracy.

The results for different reduced feature vectors evaluated in 1,000 testing set are shown in Figure 4. The F1 score for each class is plotted in Figure 4a, and all weighted metrics, F1, precision, recall, accuracy are plotted in Figure 4b. All these metrics shows 3072-dimentional (100%, original) input features performs best (accuracy: 37.3%), which is consistent with 10-fold cross validation results in Figure 3. Specifically, Figure 4a shows, among 10 type of objects, ship is the easiest to classify correctly while cat is the hardest.



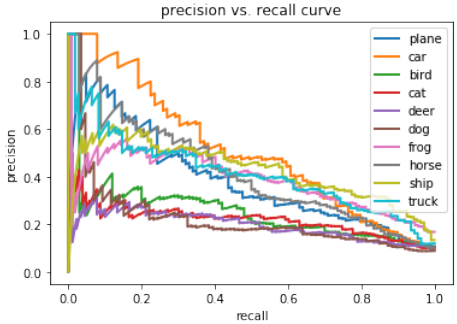

(a) F1 score for each class



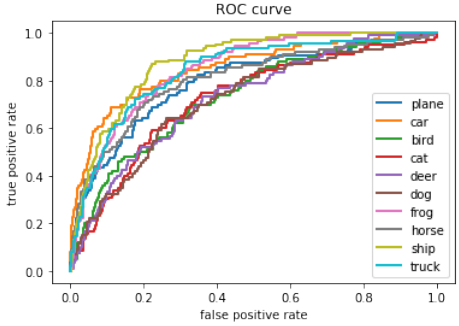

(b) F1, precision, recall, accuracy (weighted)

Figure 4: feature dimentions

The best performed linear SVM is {`kernel="linear",n_features=3072,C=0.001`}. Its precision-recall curve and ROC curve are shown in Figure 5. This reveals that physical objects, such as car, ship are less likely to misclassify than animals, like cat and dog.




(a) precision-recall curve




(b) ROC curve

Figure 5: Best performed linear SVM

To delve into the reason why certain classes are easily classify while others are not, normalized confusion matrix is plotted using `seaborn.heatmap` in Figure 6 for the best performed linear SVM. The average accuracy of physical objects {plane, car, ship, truck} is 45.0%, which is much better than the 31.2% of animals {bird, cat, deer, dog, frog, horse}. Specifically, the reason behind two least correctly identified classes, namely cat and dog, is that they are highly misclassified. For example, the percentage of cat being misidentified into dog is 19.4%, which is even higher than the correctly identified percentage of 18.4%. This can be stemmed from that these two animals are less distinctive compared with other classes.

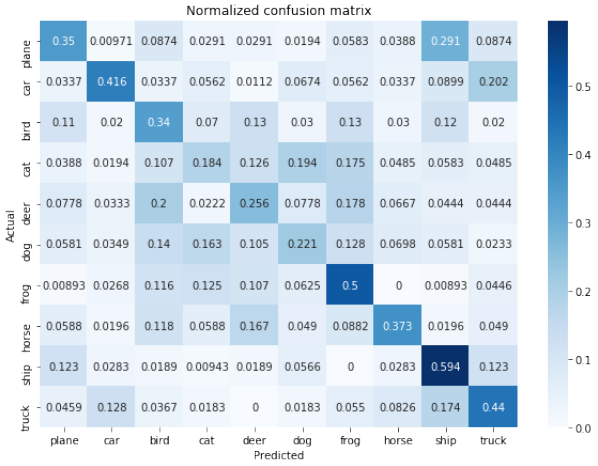



Figure 6: Confusion matrix for best performed linear SVM

## 2.3 Task 2.3

For polynomial SVM, a linear search of best parameter `degree` is implemented followed by a grid search of best combination of parameters {`C,gamma`}, both of which uses 10-fold cross validation on original (same as 3072 features) input vectors.

Firstly, the polynomial SVM is firstly validated across different degrees in the range of $[0,9]$ keeping other parameters at default. The results is shown in Figure 7, demonstrating that degree {degree=1} yields the highest accuracy.

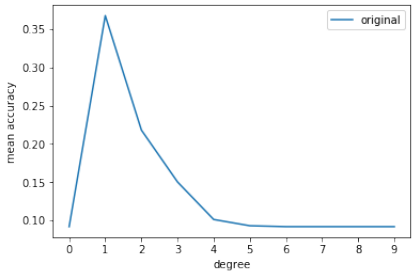
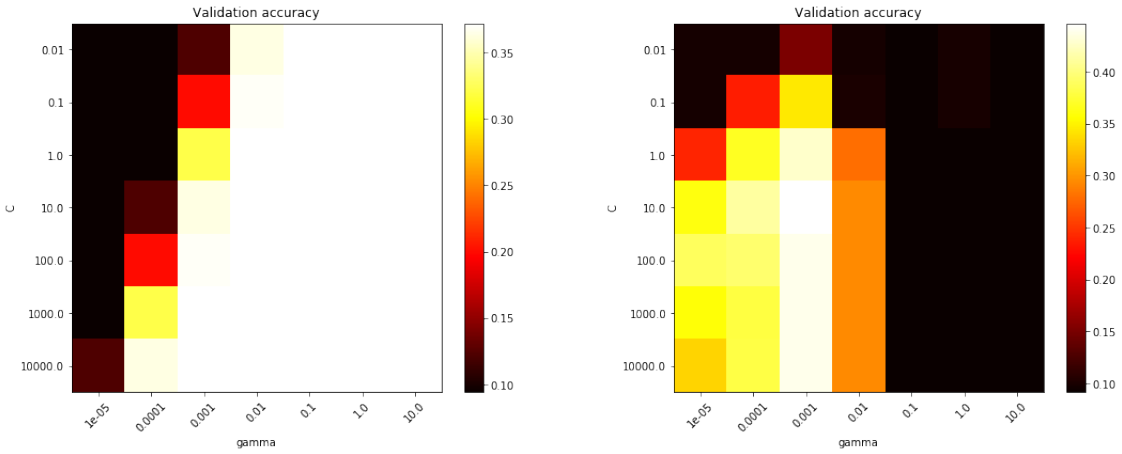



Figure 7: Degree of polynomial SVM - 10-fold CV

According to the suggested best C and gamma range for SVM in research [1], C of range $[10^{-2}, 10^{5}]$ and gamma of range $[10^{-5}, 10]$ will be used for the exhaustive grid search in the following experiments.

Then, setting {degree: 1} based on the previous result, an exhaustive search over parameters {`C,gamma`} is implemented by using `GridSearchCV`. The accuracy scores of the exhaustive 10-fold cross validation is then plotted in the heatmap shown in Figure 8a. This result together with the previous linear search gives the best polynomial SVM parameter combination {`kernel="poly",degree=1,C=1.0,gamma=0.01`}.




(a) Polynomial SVM

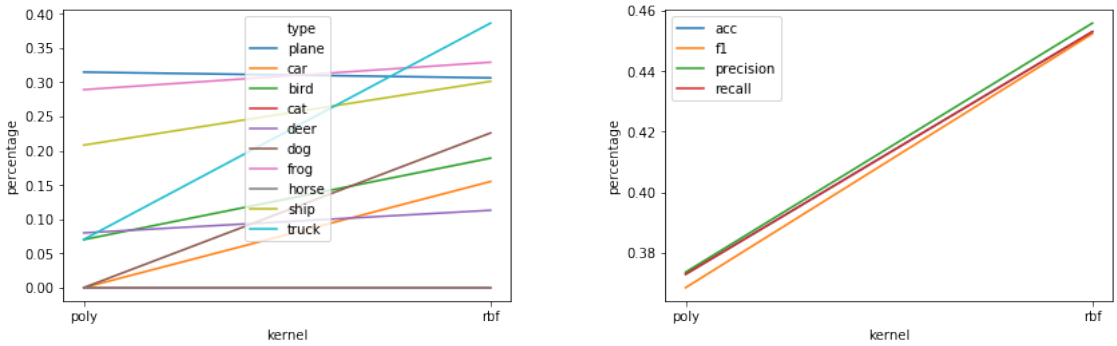
(b) RBF SVM

Figure 8: C and gamma for SVM

For RBF SVM, the same grid search on parameters {`C,gamma`} is conducted on the original input features. Its 10-fold cross validation is plotted in Figure 8b. This suggests that the best RBF SVM parameter combination is {`kernel="rbf",degree=1,C=10.0,gamma=0.001`}.

## 2.4 Task 2.4

The two best SVM of polynomial and RBF kernels are then refitted on whole training set and evaluated on testing set. In Figure 9, it is clear that RBF kernel performs better than polynomial one. Specifically, SVM with RBF kernel has an accuracy of 45.3% and F1 score of 45.2%, both of which are around 9.8% higher than the ones of polynomial SVM.




(a) F1 for each class

(b) F1, accuracy, precision, recall

Figure 9: Testing result of Polynomial and RBF SVM

# 3 Task 3

The following architectures are used for Baseline_CNN and CNN.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2d-1 | [-1, 6, 28, 28] | 456 |
| MaxPool2d-2 | [-1, 6, 14, 14] | 0 |
| Conv2d-3 | [-1, 16, 10, 10] | 2,416 |
| MaxPool2d-4 | [-1, 16, 5, 5] | 0 |
| Linear-5 | [-1, 120] | 48,120 |
| Linear-6 | [-1, 84] | 10,164 |
| Linear-7 | [-1, 10] | 850 |

Total params: 62,006
Trainable params: 62,006
Non-trainable params: 0

(a) Baseline CNN

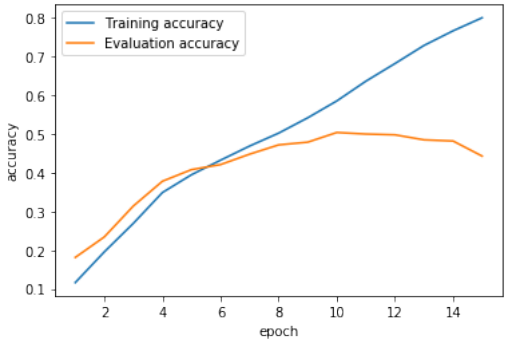
| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2d-1 | [-1, 12, 28, 28] | 912 |
| MaxPool2d-2 | [-1, 12, 14, 14] | 0 |
| Conv2d-3 | [-1, 36, 13, 13] | 1,764 |
| MaxPool2d-4 | [-1, 36, 6, 6] | 0 |
| Linear-5 | [-1, 240] | 311,280 |
| Linear-6 | [-1, 120] | 28,920 |
| Linear-7 | [-1, 84] | 10,164 |
| Linear-8 | [-1, 10] | 850 |

Total params: 353,890
Trainable params: 353,890

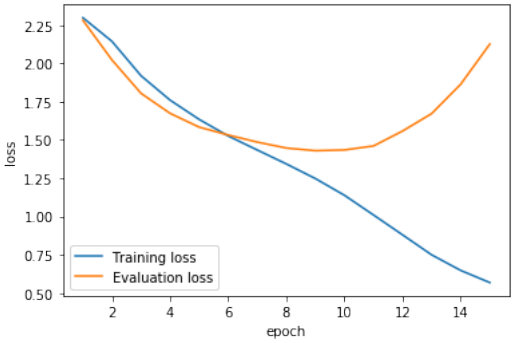
(b) CNN

Figure 10: CNN architecture

Each model is trained for 15 epochs. Models are saved during training for the purpose of evaluating loss and accuracy per epoch. In Figure 11, accuracy and loss during training/evaluation for CNN model are respectively plotted. This graph shows the effect of overfitting with increasing epoch trained, as evaluation accuracy reaching peak point and loss reaching lowest point at epoch_10.




(a) Epoch-accuracy




(b) Epoch-loss

Figure 11: CNN training and evaluation results

As can be seen in Figure 12, CNN model has a higher evaluation accuracy after epoch_8 and before epoch_15. CNN model's highest accuracy is 50.4% at epoch_10, which is 2.7% higher than the best accuracy score reached by the Baseline_CNN model.

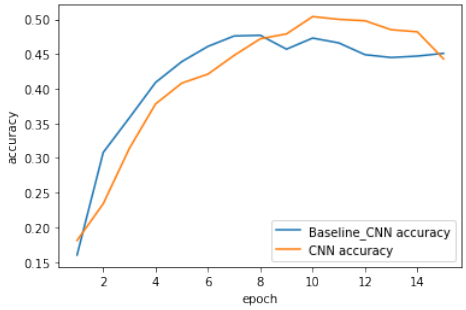
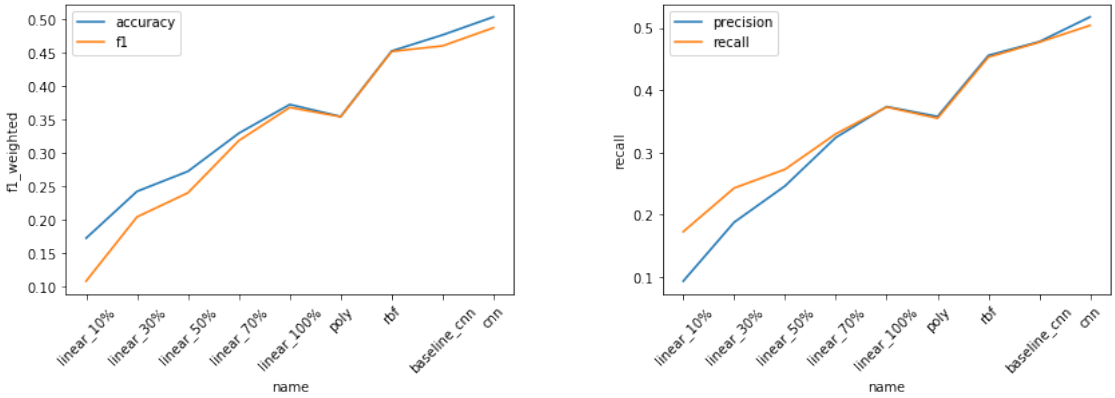



Figure 12: Evaluation accuracy for two CNN models

# 4 Task 4

The accuracy, F1, precision, recall scores for the each tasks are plotted in Figure 13. The best performed RBF SVM has the highest evaluation results (acc: 45.3%) among SVM classifiers with different kernels, input features, and parameters. However, it is still outperformed by two deep learning models, Baseline_CNN and CNN, scoring 47.7% and 50.4% on accuracy respectively.




(a) Accuracy and F1

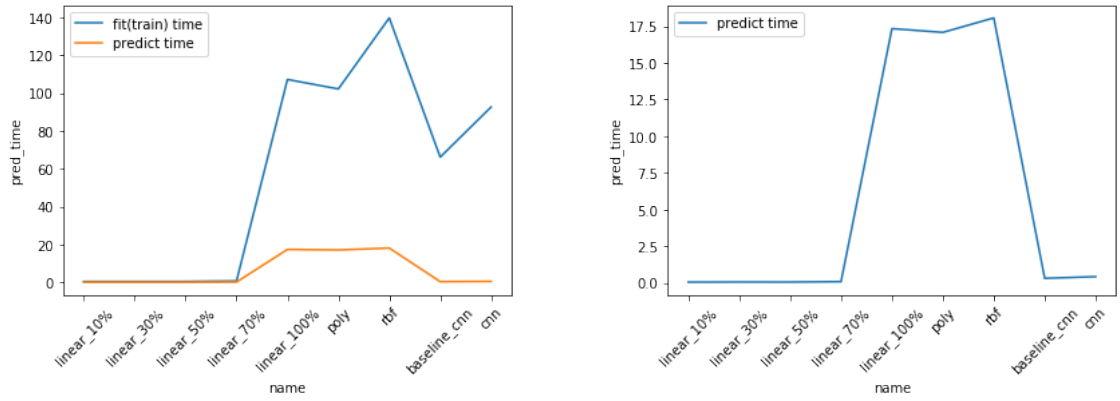
(b) Precision and recall

Figure 13: Performace comparison

In terms of methodology, the fit time (predict time) for SVM is the time used for fitting on the entire 5,000 training set (1,000 testing set), which is significantly shorter than the time spent on 10-fold cross validation, specifically on exhaustive search algorithms. On the other hand, for CNN models, train time is the time spent for 15 batches, while predict time is the time used on predicting 1,000 testing set.

Overall, RBF (139.7s), polynomial (102.1s) SVM and the one trained on original feature vectors (107.1s) has the highest training time, followed by two CNN models (66.1s, 92.5s), while reduced feature vectors (0.5s) has the lowest fitting time. Firstly, the reason why fit time for CNNs are higher than SVMs is that the former has a large amount of weights to train and it is counted for 15 epochs. The training time of SVM classifiers falls between $O(n_{features} \times n_{samples}^2)$ and $O(n_{features} \times n_{samples}^3)$, which explains why reduced feature vectors from PCA has significantly lower fit time than other SVM classifiers.

From the practical side, predict time is more important than fit/train time. In Figure 14b, CNN models spent around 0.3 seconds for predicting 1,000 training set, which is significantly better than RBF and polynomial (17.2s) models and slightly slower than SVM trained on reduced feature vectors (0.07s).




(a) Fit and predict ime

(b) Predict time

Figure 14: Time complexity comparison

# References

[1] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. “A practical guide to support vector classification”. In: (2003).